Probable Plaintext Cryptanalysis of the IP Security Protocols

Steven M. Bellovin

AT&T Labs Research
E-mail: smb@research.att.com

Abstract

The Internet Engineering Task Force (IETF) is in the process of adopting standards for IP-layer encryption and authentication (IPSEC). We describe how "probable plaintext" can be used to aid in cryptanalytic attacks, and analyze the protocol to show how much probable plaintext is available. We also show how traffic analysis is a powerful aid to the cryptanalyst. We conclude by outlining some likely changes to the underlying protocols that may strengthen them against these attacks.

1. Introduction

DES, the Data Encryption Standard [25], is a strong cipher; however, its key length is too short to provide much security against a well-financed attacker [14]. More recently, designs have been published for machines that can exhaustively search the key space in a short time for a comparatively modest investment [34].

Most such designs assume blocks of known plaintext, though at least one [33] relies on statistical properties of the underlying text. However, as will be shown, knowledge of full blocks of plaintext is not needed. The encrypted headers that are used in the forthcoming standards for IP-layer encryption and authentication (IPSEC) [3, 1, 2, 22, 21][1] provide ample *probable plaintext*. This plaintext can also be used to drive a DES-cracking engine.

A probable plaintext attack works by looking at certain bit positions for which a likely value can be predicted. Rather than looking for an exact match, though, even for those bit positions, the comparison engine counts the number of matches. Packets with more than a certain threshhold value of matches are kicked out for further analysis by a second-stage engine; this could involve more probable plaintext, semantic consistency checks, even (ultimately) human analysis.

It is not necessary for all matches to be within a single ciphertext block. However, since the cost of a decryption is roughly proportional to the number of decryption operations, it is desirable to find single blocks with a high amount of probable plaintext.

Normally, trial decryptions will be preceded by an data gathering phase. This phase uses traffic analysis, packet length, auxiliary information determined by other means including conventional intelligence activities, etc., to determine the likely probable plaintext patterns.

Section 2 describes our notation and the relevant properties of the encryption modes used. Section 3 describes the architecture of IPSEC. A detailed analysis of the probable plaintext, using one and two packets of ciphertext, is given in Sections 4 and 5. Some of the attacks described depend on the ability of the attacker to identify particular conversations; how this can be done is sketched in Section 6. We conclude with a discussion of possible defenses (Section 7) and a set of recommendations (Section 9).

2. Properties of Encryption Modes

Our primary focus here is DES used in *cipher block chaining* mode (CBC) [26]. For this form of attack, stream ciphers are essentially equivalent to a CBC-mode block cipher where the initialization vector is known; this has some minor implications for the single-packet attack.

The discussion below focuses on aspects of interest to us. More detailed information on the properties of these and other cipher modes can be found in [32].

2.1. Notation

We use $C_i = K[P_i]$ to mean "ciphertext C_i results from the encryption of plaintext P_i using key K. The corresponding decryption is written $P_i = K^{-1}[C_i]$. The symbol $\oplus$ denotes bitwise exclusive-OR.

[1] These RFCs are obsolete, but at press time have not yet been replaced by newer versions.

If a number is written with a subscript, that subscript denotes the base; unsubscripted numbers are in base 10.

2.2. Cipher Block Chaining

CBC encryption [26] operates by encrypting the exclusive-OR of each plaintext block and the previous ciphertext block:

$$C_i = K[P_i \oplus C_{i-1}].$$

To encrypt the first plaintext block, C_0 is set to the *initialization vector* (IV). IVs may be agreed upon in advance, transmitted encrypted, or transmitted in the clear. Using non-constant IVs is sometimes recommended, in order to disguise common prefixes. In the draft under discussion here, a constant IV, derived from the keying material, is used for all packets, in either direction. The use of a replay counter serves to disguise block prefixes.

Decryption is the inverse operation:

$$P_i = C_{i-1} \oplus K^{-1}[C_i].$$

To encrypt data that is not a multiple of the underlying cipher's block size, some sort of padding and length information must be added. There are a number of different techniques that may be used; none of the straight-forward schemes add much to the security of the encryption.

If the IV is unknown, it is impossible to decrypt the first block. In effect, a secret IV acts as a second key, but only for processing the first block.

All subsequent blocks can be decrypted given knowledge of the key and the preceding ciphertext (not plaintext) block. That is, a substring $\langle C_i, \ldots, C_j \rangle$ is a valid CBC encryption of $\langle P_i, \ldots, P_j \rangle$, with the IV set to C_{i-1}.

3. The IPSEC Encryption Protocols

The packet layout for the current draft specification for the Encapsulating Security Payload (ESP) (`draft-ietf-ipsec-esp-des-md5-03.txt`) is shown in Figure 1. The first 32 bits contain the SPI (Security Parameter Identifier). The SPI serves as an index to the key, the IV, etc. The granularity of an SPI is not defined by the standard; it may be for a single conversation, or it may cover all traffic between a pair of hosts. Since the SPI is used to find the decryption key, it is transmitted in the clear; the remainder of the packet is encrypted using DES in CBC mode.

The replay counter is initialized from the keying material when the security association is created. It is not allowed to wrap around; a new key must be negotiated before 2^{32} packets are transmitted.

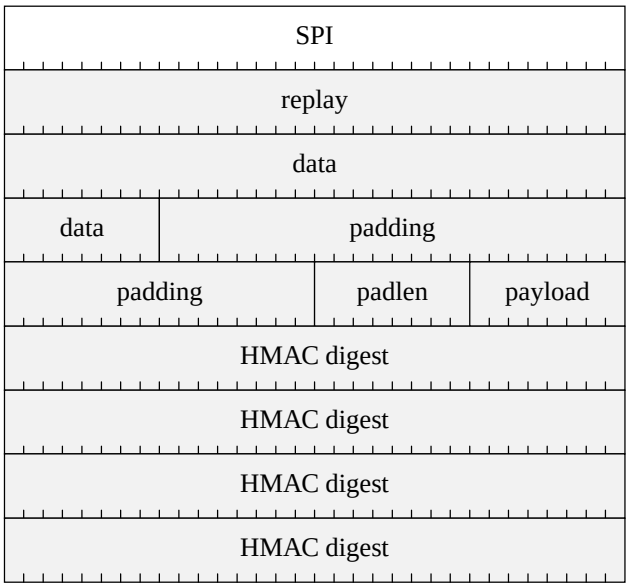

Figure 1. Format of ESP packets. The shaded portion of the packet is encrypted using DES in CBC mode.

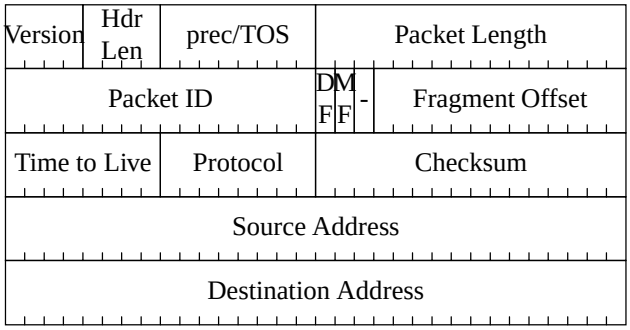

Figure 2. Format of the IP header.

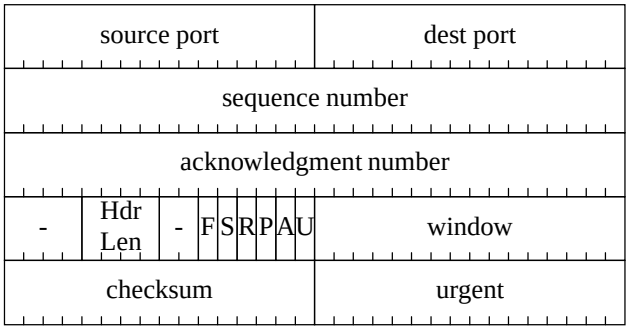

Figure 3. Format of the TCP header.

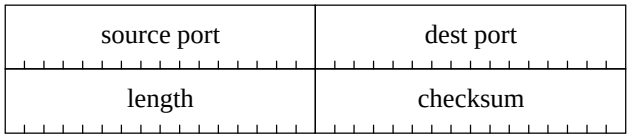

Figure 4. Format of the UDP header.

Encryption may be host-to-host, host-to-firewall, or firewall-to-firewall. In the latter two cases, *tunnel mode* encryption is used; the payload of the encryption is an entire IP packet [29], including the real IP header (Figure 2). For host-to-host encryption, tunnel mode may be used; more likely, the encrypted data starts with the TCP [30] or UDP [28] headers (Figures 3 and 4).

Because CBC encryption operates on 8-byte blocks, short packets must be padded. Up to 255 bytes of random padding may be used; however, the amount must bring the total length to 6 bytes more than a multiple of 8. The padding is followed by a single byte that tells how much padding was used. This is followed by a payload type byte; it identifies the header type of the encrypted data. If tunnel mode is not used, this value is (conceptually) inserted into the IP header when the ESP control data is deleted.

The last portion of the encrypted packet is the authentication field. The authentication field is calculated using the HMAC transform [4], and is based on a negotiated secret key. Authentication is mandatory, to avoid some of the attacks described in [7].

4. Single-Packet Attacks

In a single-packet attack, trial decryptions are done on one packet at a time. The analysis differs depending on whether or not tunnel mode is used. It may be known *a priori*, because of the presence of firewall routers; if not, the packet length may be useful in making a determination. Measurements have shown that 30-40% of all packets are 40-byte TCP ACK packets [12, 23]. If random padding lengths are not used; the size will show through directly; if they are used, analysis of the distribution of lengths should yield sufficient information.

Depending on whether the IV is known, we may or may not be able to attack the first block. The current draft indicates that the IV is not known, which is just as well; the content of the block can sometimes be predicted with great accuracy.

The first word is the replay counter. In early versions of the specification, the replay counter was defined as starting at one; if we could intercept packets from the beginning of the association, we would know at least 30 bits, and possibly all 32. Even with a comparatively late interception, we could probably assume that the high-order 20-24 bits are zero, especially if security associations are short-lived. (There is an interesting tradeoff here. Conventional cryptographic wisdom calls for limiting the amount of plaintext encrypted under any one key. But too stringent a limit increases the predictability of the value of the replay counter. As we shall see, there are other such tradeoffs as well.)

More recent versions of the specification derive the starting value from the keying material. This blocks single-packet attacks.

4.1. Probable Plaintext in the IP Header

Our analysis must now be done for each possible header. If tunnel mode is used, the IP header is next. The first word of it turns out to be very predictable. The version number is always 4_{16}; the header length is almost always 5_{16}, and the precedence/type-of-service field is generally 10_{16}. (Some implementations will set one of the type-of-service bits; if the likelihood of this bit being set cannot be determined by traffic analysis, we lose at most one bit of predictability.)

We also know a lot about the packet length, simply by seeing how long the encrypted packet is. If random padding lengths are not used; we can calculate all but the low-order few bits. If random lengths are used, we can rely on traffic analysis to pick out the ACK packets, for which the IP length will always be 28_{16}.

Under favorable circumstances, then, the first ciphertext block of a tunnel mode packet contains about 60 bits of predictable plaintext. If the IV is known, this block would be a prime target for a cryptanalysis device.

The next block is of less use to the attacker, yielding only 24-28 bits of probable plaintext. The packet ID and checksum are effectively random numbers. The fragment offset and flag fields, though, are generally all 0; the protocol identifier is almost certainly either 6_{16} for TCP or 11_{16} for UDP (and traffic analysis will tell us which), and the time-to-live field will depend only on the behavior of the sending host's protocol stack and on its distance in hops from the encryptor. These may be known, at least to within a small margin of error; if so, we can estimate values for the high-order few bits.

The remaining fields of the IP header are the source and destination addresses. In host-to-firewall mode, one of the cleartext IP addresses will match the encrypted copy, giving us 32 bits; if tunneling is used even though we are in host-to-host mode, we know all 64 bits.

The more likely use of tunnel mode is for firewall-to-firewall encryption. How many of the address bits are predictable will depend entirely on the circumstances. If valid, assigned Internet addresses are used behind the firewall, and these are known to the attacker (perhaps from unencrypted connections from inside hosts to other machines on the outside), 16-24 bits per address can be predicted. The use of CIDR block addressing [16] makes this even more likely. It

would seem desirable, then, to use arbitrary addresses for machines behind the firewall, and rely on application gateways [11] or network address translators [15] to conceal this data. A note of caution is indicated here, though; addresses can leak in many ways, and it is hard to close all such channels.

4.2. Probable Plaintext in the TCP Header

Because the TCP and UDP headers can follow either the replay counter or an IP header, we cannot analyze them in terms of ciphertext blocks; the alignment will differ. Accordingly, we will speak in terms of fields or words.

The first word contains the source and destination port numbers. In general, the client's port number is unpredictable, and should be considered random; often, however, the server's port number can be deduced from traffic characteristics (Section 6). If the determination can be made, 16 bits of probable plaintext are available.

Sequence numbers are usually random. Under certain circumstances, however, they are quite predictable [24, 6]; if the attacker can establish its own connection to the source machine at roughly the same time as the first packet of the intercepted connection, 28-32 bits of the sequence number can be predicted. Initial sequence number randomization [5] would be a strong defense here.

The acknowledgment field is the reflection of the other party's sequence number; as such, it shares the same constraints on predictability. At one crucial point, though—the initial SYN packet sent by the client to open the connection—the entire field is zero. If this packet can be identified, 32 bits of known plaintext are available.

The next word, containing a length field, flags, and the window size, is often completely predictable. The length is almost always 5_{16}, the SYN bit is set at the start of a conversation but not otherwise, the FIN bit at the conclusion, the ACK bit is always set except in the first packet, the RST bit is rarely used in any conversation of interest, the URG bit is seldom set, and the setting of the PSH bit can be determined from a knowledge of the traffic pattern and of the implementation sending it. The window size is somewhat harder; however, given that most conversations are unidirectional at any given time, the sending side is usually advertising a full window whose size is in general a characteristic of the particular stack. Even without that, most TCP implementations use window sizes that are powers of two; an assumption that the field will be all 0s will have an error in exactly one bit position.

As with IP, the checksum field is unpredictable; the urgent pointer, however, is almost always zero, and hence is predictable.

Under reasonable assumptions—that traffic analysis will supply the destination port, and that we can identify the first packet of a conversation—the TCP header therefore has 88 bits of probable plaintext, with a small uncertainty about the exact window size. Furthermore, a single ciphertext block will contain either a $\langle 32, 32 \rangle$ pair or a $\langle 32, 16 \rangle$ pair of words.

4.3. Probable Plaintext in the UDP Header

The analysis of the UDP header is similar, though of course simpler. Again, we can deduce the server's port number by traffic analysis; the length field can be approximated from the total packet length. A fair estimate would be 28 bits of probable plaintext in the UDP header, which is likely too short without a lot of traffic.

5. Two-Packet Attacks

We can obtain even more probable plaintext by analyzing pairs of packets from the same conversation. Since each packet will have a different replay counter, the first block of ciphertext will always be different; this in turn will propagate to all other blocks of ciphertext in the message. If some field should be constant in two different packets, and decrypts the same way each time, we have good reason to believe that the key was correct for both packets.

A two-packet cryptanalysis device has one key generator, but two decryption engines that operate in parallel. The output of the two engines is compared using the same probable plaintext techniques discussed earlier. In other words, we are using two decryption devices for each candidate key, rather than one; thus, the cost of the machine will roughly double for a given level of performance.

The benefits of two-packet attacks can be seen most easily for the source and destination addresses fields in the IP header. As noted above, if tunnel mode is used for firewall-to-firewall encryption, the attacker has little knowledge of what those fields should be. But if two packets from the same conversation are decrypted, the two fields will match each other. Similarly, the TCP and UDP port numbers can be compared this way.

Fields that change slowly can also be compared, though not with as much precision, by taking advantage of the limited leftward propagation of carries when adding small values to a counter.

Counter Theorem: If a value 2^k is added to a uniformly distributed random n-bit number M, the probability that any of bits 0 through i of M are changed is $1/2^{n-i-1-k}$, for $i < n - k$. The formal proof, by induction on i, is left as an exercise for the reader; informally, a bit is changed if and only if all of the bits to its right are 1s, up through the bit position where the increment takes place.

Consider the sequence number field. If a 512-byte packet is sent—common for many implementations of TCP—the

sequence number of the following packet will be incremented by 512, or 2^9. By the counter theorem, there is a probability of .97 that bits 0-17 of the sequence number will be unchanged in the second packet. For that matter, we also know that bits 23-31 will be completely unchanged if exactly 512 bytes are sent, giving us 27 bits of plaintext with high probability.

The acknowledgment field can be treated similarly; however, because multiple acknowledgments are often compressed into a single reply, our bound on the number of unchanged bits is somewhat looser.

If we see a rapid string of packets from one conversation, the counter theorem can be also be applied to the IP packet ID and to the replay counter. The ID field is always incremented by one for each packet in a security association, and the ID field is generally incremented for each packet sent to any destination. If a group of packets is transmitted in a burst, without other processes on the sending machine intervening, we can assume that the packets in the burst will receive consecutive numbers.

Finally, under the right conditions we can apply this theorem to the TCP client port number field. Many Web pages contain embedded images; each of these is retrieved by a separate TCP connection. Again, if these connections are closely spaced in time, they will receive consecutive port numbers.

Together, all of these heuristics boost the amount of probable plaintext considerably. Under a two-packet attack, the IP header, 160 bits long, has about 127 bits of probable plaintext, and the TCP header has about 124 bits predictable out of 160.

That there is considerable redundancy here is not surprising. Indeed, the same observation was made, albeit in a rather different context, in the design of PPP header compression [18]. When predictable header fields are added to the redundancy in the user's data, it becomes apparent that exhaustive search cryptanalysis is quite feasible even without known plaintext.

6. Traffic Analysis

Probable plaintext attacks are not carried out on a whim. They require expensive, special-purpose hardware. An enemy who builds such a device will do other sorts of monitoring to prepare the attack.

One form of monitoring is traffic analysis. Under the right conditions, traffic analysis can reveal a lot about a conversation, and provide information that will aid in cryptanalysis. This is most easily seen by looking at TCP.

The open sequence in TCP consists of three messages. The first is a 40 or 44-byte message from the client to the server, containing an IP header, a TCP header with the ACK bit off and the SYN bit on, the acknowledgment field set to 0, and possibly a TCP option specifying a maximum segment size. (Some implementations will also send the RFC 1323 [9] options; these seem to be rare at present.)

The second message is similar, though the ACK bit is set and the acknowledgment field is non-zero. The third message is a simple 40-byte message with ACK on and SYN off. It is generally followed in short order by a message from one side containing a few dozen to a few hundred bytes of data.

The sequence described above is easily recognizable, especially if there has been no other recent traffic between the two hosts or firewalls. If per-connection keying is used, recognition is even easier, of course.

Different protocols have their own characteristic traffic patterns. For example, SMTP [31] has a series of short data-bearing packet exchanges between the two sides, followed by a longer message from the client, and another set of brief exchanges. HTTP [8] exchanges consist of a few hundred bytes sent in one direction, followed by at least several hundred bytes in the other direction. Also, many real HTTP exchanges consist of several such sessions opened in short order. Further examples are left as an exercise.

Packet interarrival times can also be used. Some years ago, a phenomenon known as "packet trains"—bursts of packets from a single stream—was identified [19], though that applies more to local traffic than to wide-area traffic. Statistical studies have been done as well; see, for example, [27, 10]. The latter paper showed the distribution of packet sizes and interpacket arrival times for some different protocols; the differences are striking.

Perhaps more importantly, [10] also shows a very low probability of more than one conversation between any given pair of hosts. Even the number of simultaneous conversations between pairs of networks is quite low. To be sure, their data is pre-Web, but even Web page fetches typically represent related conversations.

We know of no published work on traffic analysis of Internet conversations. It would be useful to attempt such measurements, using existing packet header data to assess the success or failure of the classification.

7. Defenses

7.1. Removing Redundancy

To defend against probable plaintext attacks, one needs to reduce the predictability of the header fields. In some cases, this is easy; in other cases, it cannot be done without changes to the underlying protocols.

The simplest change is to avoid exposure of the IV. As of this writing, the IV is not sent in the clear; this is probably wise. But if a key for one direction is recovered by a two-packet attack, probable plaintext techniques can be used to

recover the IV; this in turn may provide probable plaintext that can be used to attack the key used for the other direction.

Given this, it was wise to change definition of the replay counter. Instead of starting it at zero, it now starts some random value derived from the keying material. This adds at most a single subtraction to input processing: the starting value must be subtracted from the reply counter mod 2^{32} before checking for wraparound. Another big improvement is to avoid use of host-to-host tunnel mode. For host-to-firewall mode, where tunnel mode must be used, a random value can be substituted for the encrypted copy of the exposed IP address; the receiving machine should know from the key negotiation that this is taking place, and substitute in the proper value. (To avoid spoofing, though, it may be wise to do authentication calculations on the correct address.)

Little can be done about the other fields in the IP header. The protocol field will generally specify TCP or UDP, fragment offsets will almost always be 0, etc. If desired, the packet ID could be selected from a permutation table; this would provide some defense against two-packet attacks on the IP header.

The semantic properties of the TCP header are even more difficult to hide. Minor changes are easy, such as sending random values for the acknowledgment field in the initial SYN packet, or sending random values for the urgent pointer at times when the URG bit is not set. But it is hard to see how to hide, say, the constant value of the acknowledgment field in packets carrying bulk data in one direction.

7.2. Compression

Compression is often touted as a cure for excess redundancy [33]. An ordinary compression function is theoretically inadequate, though it may provide some benefits in the short term.

The problem is that the decryption engine could easily do a trial decompression as well, before looking for the probable plaintext. For now, the extra logic circuits and time required may make this attack infeasible in practice; soon, though, advances in hardware design will negate the defender's advantage.

A more promising approach might be to use a keyed compression function. For example, the compression dictionary could be modified based on the keying material.

A different approach to compression might be to use semantic knowledge, along the lines of PPP header compression [18]. For example, it may be possible to send abbreviated sequence and acknowledgment fields. If per-connection keying is used, IP addresses and port numbers are implicit in the security association and need not be sent. The replay counter is more troublesome; more or less by definition, it can't be abbreviated, and hence remains vulnerable to a two-packet attack. Keeping the IV secret, and using different IVs in different directions, should help.

7.3. Avoiding Traffic Analysis

The best defense against two-packet attacks (and against some forms of one-packet attacks) is to deny the enemy information about which packets belong to which conversation. Unfortunately, per-connection keying—recommended above as well as in [7]—is the easiest tipoff for the attacker.

Timing correlations are also useful clues. It is likely to be hard to detect such things on the long-haul backbone nets, but the use of link encryption may be appropriate on the line from an individual organization to its ISP. Other forms of multiple encryption help as well; for example, firewall-to-firewall encryption can be seen as a complement to host-to-host encryption. We face a cryptographic conundrum here. On the one hand, encrypting data from many streams at once helps defend against traffic analysis. On the other hand, it is generally thought unwise to encrypt too much data with one key, or to use the same key for data at different sensitivity levels.

To some extent, packet sizes can be obscured by dummy traffic, or by non-uniform (or even keyed) distributions of the padding length. Both are unfriendly to the infrastructure; the Internet is congested enough as is, without being asked to carry unproductive data. To be sure, the padding length byte itself can be considered as probable plaintext, but using different amounts of padding would not seem to matter.

8. Related Work

Probable plaintext cryptanalysis is not new. Perhaps the most noteworthy example is the successful Allied attack on Enigma during World War II [13, 20, 17]. For example, weather forecasts often began *Wettervorhersage Deutsche Bucht* ("weather forecast German Bight") [17, p. 53]. Similarly, the cryptanalysts often used messages encrypted in

Table 1. Summary of probable plaintext under single- and double-packet attacks.

	Single	Double
IP	54–58*	127
TCP	88	124
UDP	28	46

*If tunnel mode is used, the IP header will have 32 or 64 more bits of probable plaintext.

both Enigma and a simpler system; solving one provided a "crib" for the other. On occasion, they even resorted to chosen plaintext attacks; new minefields were regularly reported in several different cryptosystems [20, p. 144].

There were analogues to double-packet cryptanalysis as well. The original keying practice called for two encryptions of the message's initial rotor settings; while the cryptanalysts did not know what these settings were, they did know that the pairs had to match [13].

The use of traffic analysis as an aid to cryptanalysis is described in [20, p. 98], among other places. One British cryptanalyst noted that the use of the same "keys" (actually, key families, in modern terminology) indicated a common command structure, which in turn suggested common addressees.

9. Recommendations

We have shown here how easy it is to find probable plaintext in the headers (Table 1). For sensitive material, countermeasures must be taken.

The simplest defense, of course, is to avoid use of weak ciphers. There is little doubt that DES is inadequate against a serious opponent. That security systems based on it should be vulnerable is not surprising; what we have simply analyzed exactly how to attack one instantiation.

There are a few spots where minor changes to TCP implementations would help, such as using random values in "don't care" fields. But these are of lesser value; the main points of vulnerability—the acknowledgment and sequence number fields in most packets—cannot be disguised in this fashion.

Measures to thwart traffic analysis are useful against two-packet attacks. As noted, encryption at the firewall is a useful adjunct to host-based encryption.

For the longer term, work on keyed or semantic compression should be undertaken. There is a need for an IPSEC compression transform for use over modems; we recommend that due attention be given to probable plaintext attacks when designing it.

10. Acknowledgments

Kim Claffy, Lixia Zhang, Greg Minshall, and Ramón Cáceres provided useful pointers, data, and references on traffic characteristics.

References

[1] R. Atkinson. IP authentication header. Request for Comments (Proposed Standard) RFC 1826, Internet Engineering Task Force, Aug. 1995.

[2] R. Atkinson. IP encapsulating security payload (ESP). Request for Comments (Proposed Standard) RFC 1827, Internet Engineering Task Force, Aug. 1995.

[3] R. Atkinson. Security architecture for the internet protocol. Request for Comments (Proposed Standard) RFC 1825, Internet Engineering Task Force, Aug. 1995.

[4] M. Bellare, R. Canetti, and H. Krawczyk. Keying hash functions for message authentication. In *Advances in Cryptology: Proceedings of CRYPTO '96*, pages 1–15. Springer-Verlag, 1996.

[5] S. Bellovin. Defending against sequence number attacks. Request for Comments (Informational) RFC 1948, Internet Engineering Task Force, May 1996.

[6] S. M. Bellovin. Security problems in the TCP/IP protocol suite. *Computer Communications Review*, 19(2):32–48, April 1989.

[7] S. M. Bellovin. Problem areas for the IP security protocols. In *Proceedings of the Sixth Usenix* UNIX *Security Symposium*, pages 205–214, July 1996.

[8] T. Berners-Lee, R. Fielding, and H. Nielsen. Hypertext transfer protocol — HTTP/1.0. Request for Comments (Informational) RFC 1945, Internet Engineering Task Force, May 1996.

[9] D. Borman, R. Braden, and V. Jacobson. TCP extensions for high performance. Request for Comments (Proposed Standard) RFC 1323, Internet Engineering Task Force, May 1992. (Obsoletes RFC1185).

[10] R. Cáceres, P. B. Danzig, S. Jamin, and D. J. Mitzel. Characteristics of wide-area TCP/IP conversations. In *Proceedings of SIGCOMM '91*, 1991.

[11] W. R. Cheswick and S. M. Bellovin. *Firewalls and Internet Security: Repelling the Wily Hacker*. Addison-Wesley, Reading, MA, 1994.

[12] K. Claffy. WAN packet size distribution, 1996. `http://www.nlanr.net/NA/Learn/packetsizes.html`.

[13] C. A. Deavours and L. Kruh. *Machine Cryptography and Modern Cryptanalysis*. Artech House, Norwood, MA, 1985.

[14] W. Diffie and M. E. Hellman. Exhaustive cryptanalysis of the NBS data encryption standard. *Computer*, 10(6):74–84, June 1977.

[15] P. Francis and K. Egevang. The IP network address translator (nat). Request for Comments (Informational) RFC 1631, Internet Engineering Task Force, May 1994.

[16] V. Fuller, T. Li, J. Yu, and K. Varadhan. Classless interdomain routing (CIDR): an address assignment and aggregation strategy. Request for Comments (Proposed Standard) RFC 1519, Internet Engineering Task Force, Sept. 1993. (Obsoletes RFC1338).

[17] F. Hinsley and A. Stripp, editors. *Codebreakers: The Inside Story of Bletchley Park*. Oxford University Press, 1993.

[18] V. Jacobson. Compressing TCP/IP headers for low-speed serial links. Request for Comments (Proposed Standard) RFC 1144, Internet Engineering Task Force, Feb. 1990.

[19] R. Jain and S. Routhier. Packet trains—measurements and a new model for computer network traffic. *IEEE Journal of Selected Areas in Communications*, AC-4(6):986–995, September 1986.

[20] D. Kahn. *Seizing the Enigma: The Race to Break the German U-Boat Coes, 1939–1943*. Houghton Mifflin, Boston, 1991.

[21] P. Metzger, P. Karn, and W. Simpson. The ESP DES-CBC transform. Request for Comments (Proposed Standard) RFC 1829, Internet Engineering Task Force, Aug. 1995.

[22] P. Metzger and W. Simpson. IP authentication using keyed MD5. Request for Comments (Proposed Standard) RFC 1828, Internet Engineering Task Force, Aug. 1995.

[23] G. Minshall. Byte size distribution data, 1995. `http://www.nlanr.net/NA/Learn/Gm/pktsizes.html`.

[24] R. T. Morris. A weakness in the 4.2BSD UNIX TCP/IP software. Computing Science Technical Report 117, AT&T Bell Laboratories, Murray Hill, NJ, February 1985.

[25] NBS. Data encryption standard, January 1977. Federal Information Processing Standards Publication 46.

[26] NBS. DES modes of operation, December 1980. Federal Information Processing Standards Publication 81.

[27] V. Paxson and S. Floyd. Wide-area traffic: The failure of Poisson modeling. In *Proceedings of SIGCOMM '94*, pages 257–268, London, UK, 1994.

[28] J. Postel. User datagram protocol. Request for Comments (Standard) STD 6, RFC 768, Internet Engineering Task Force, Aug. 1980.

[29] J. Postel. Internet protocol. Request for Comments (Standard) RFC 791, Internet Engineering Task Force, Sept. 1981. (Obsoletes RFC0760).

[30] J. Postel. Transmission control protocol. Request for Comments (Standard) STD 7, RFC 793, Internet Engineering Task Force, Sept. 1981.

[31] J. Postel. Simple mail transfer protocol. Request for Comments (Standard) STD 10, RFC 821, Internet Engineering Task Force, Aug. 1982. (Obsoletes RFC0788).

[32] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, New York, second edition, 1996.

[33] D. A. Wagner and S. M. Bellovin. A programmable plaintext recognizer, 1994. ftp://ftp.research.att.com/dist/smb/recog.ps.

[34] M. J. Wiener. Efficient DES key search. Technical Report TR-244, School of Computer Science, Carleton University, Ottawa, Canada, May 1994. Presented at the Rump Session of Crypto '93.